“Academia” Modeling Example

- We will model an academic enterprise expressing relationships between
  - People
    - Faculty
    - Students
      - Graduate
      - Undergraduate
    - Instructors – which can be grad students or faculty
  - Courses
  - Academic departments
  - Personal ID numbers

How should we model these basic domains in Alloy?
Strategy

- Build and validate your model incrementally
  - Start with basic sets and relations
  - Add basic constraints
  - Instantiate the model and study the results
  - Probe the model with assertions
- Add groups of features at a time
  - New sets and relations
  - New constraints
  - Confirm previous assertions
  - Probe new features with assertions

Basic Components

- People
  - Students: Undergrads and Grads
  - Instructors: Faculty and Grads
- Courses
- Relationships
  - One instructor teaches a course.
  - One or more students are taking a course.
  - Students can be waiting for a course.
Academia Domains

model Academia {
    domain {Person, Course}

    state {
        partition Faculty, Student : Person
        partition Undergrad, Graduate : Student
        Instructor : Person
        ...
    }
}

Note that we cannot specify here that instructors can only be grad students or faculty. We will do that later in an invariant schema.

Academia Associations

state {
    ...
    teaches (~taughtby) : Instructor! -> Course
    taking (~enrolled) : Student+ -> Course
    waitingfor (~waitlist) : Student -> Course
}

Note: “per” = “for each”
Academia Constraints

- All instructors are either faculty or graduate students.
  - Couldn't be expressed in set definition
- No one is waiting for a course unless someone is enrolled.
- No graduate student teaches a course that they are enrolled in.

inv {
  // All instructors are either Faculty or Graduate Students
  all i : Instructor | i : Faculty || i : Graduate

  // no one is waiting for a course unless someone is enrolled
  all c : Course | some c.waitlist -> some c.enrolled

  // no graduate student teaches a course that they are enrolled in
  all g : Graduate |
    g : Instructor -> no (g.teaches & g.taking) + (g.teaches & g.waitingfor)
}
There is a graduate student whom is an instructor.

There are at least...
  - Two courses and
  - Three undergraduates
Academia Assertions

- No instructor is on the waitlist for a course that he/she teaches.
- No student is enrolled and on the waitlist for the same course.

// No instructor is on the waitlist for a course that he/she teaches
assert NoWaitingTeacher {
    all c : Course | no (c.taughtby & c.waitlist)
}

// No student is enrolled and on the waitlist for the same course
assert NoEnrolledAndWaiting {
    all c : Course | no (c.enrolled & c.waitlist)
}
For you to do (pause here)

- Load academia-1.all
- By looking at various sample instances, do you consider the model to be underconstrained in any way?
- With realism conditions enabled, do any instances exist in the default scopes?
  - Manipulate the scopes as necessary to obtain an instance under the realism conditions.
- Check assertions

Default Instance

Analyzing state ...
Scopes: Course(3), Person(3)
Instance found:
Domains:
  Course = {C0,C1,C2}
  Person = {P0,P1,P2}
Sets:
  Faculty = {P1,P2}
  Graduate = {P0}
  Instructor = {P1,P2}
  Student = {P0}
  Undergrad = {}
Relations:
  enrolled = {C0 -> {P0}, C1 -> {P0}, C2 -> {P0}}
  taking = {P0 -> {C0,C1,C2}}
  taughtby = {C0 -> {P2}, C1 -> {P1}, C2 -> {P1}}
  teaches = {P1 -> {C1,C2}, P2 -> {C0}}
  waitingfor = {P0 -> {C0,C1,C2}}
  waitlist = {C0 -> {P0}, C1 -> {P0}, C2 -> {P0}}

Need to relate enrollment and waiting lists
Realism constraints

Analyzing Realism ...
Scopes: Course(3), Person(3)  
No instances exist in this scope

Domains:
- Course = \{C0,C1,C2\}
- Person = \{P0,P1,P2,P3\}

Sets:
- Faculty = \{
- Graduate = \{P0\}
- Instructor = \{P0\}
- Student = \{P0,P1,P2,P3\}
- Undergrad = \{P1,P2,P3\}

Relations:
- enrolled = \{C0 -> \{P3\}, C1 -> \{P3\}, C2 -> \{P1,P2,P3\}\}
- taking = \{P1 -> \{C2\}, P2 -> \{C2\}, P3 -> \{C0,C1,C2\}\}
- taughtby = \{C0 -> \{P0\}, C1 -> \{P0\}, C2 -> \{P0\}\}
- teaches = \{P0 -> \{C0,C1,C2\}\}
- waitingfor = \{P0 -> \{C2\}, P2 -> \{C0,C1,C2\}, P3 -> \{C0,C1\}\}
- waitlist = \{C0 -> \{P2,P3\}, C1 -> \{P2,P3\}, C2 -> \{P1,P2\}\}

3 undergrads
1 grad student whose an instructor

Counter-example

Analyzing NoEnrolledAndWaiting ...
Scopes: Course(3), Person(3)  
Counterexample found:

Domains:
- Course = \{C0,C1,C2\}
- Person = \{P0,P1,P2\}

Sets:
- Faculty = \{P2\}
- Graduate = \{P0,P1\}
- Instructor = \{P0,P1\}
- Student = \{P0,P1\}
- Undergrad = \{

Relations:
- enrolled = \{C0 -> \{P0\}, C1 -> \{P0\}, C2 -> \{P0\}\}
- taking = \{P0 -> \{C0,C1,C2\}\}
- taughtby = \{C0 -> \{P1\}, C1 -> \{P1\}, C2 -> \{P1\}\}
- teaches = \{P1 -> \{C0,C1,C2\}\}
- waitingfor = \{P0 -> \{C2\}\}
- waitlist = \{C2 -> \{P0\}\}
Extension 1

- Add an attribute for students
  - Unique ID numbers
  - This requires a new domain
- Add student transcripts
- Add prerequisite structure for courses

New Relations

```plaintext
state {
  ...
  prerequisites : Course -> Course
  transcript : Student -> Course
  id : Student! -> Id!
}
```

Exactly one student per id
Exactly one Id per student
New Constraints

- No uniqueness constraint for student Id’s is required (enforced by multiplicities)
- A student can only take a course for which they have already taken the prereqs
- A course does not have itself as a prerequisite.
- Realism: that there exists a course with prerequisites that someone is enrolled in

Academia Constraints

```plaintext
inv {
  
  // A student can only take courses if they have taken the prerequisites
  all s : Student | s.taking.prerequisites in s.transcript

  // There are no cycles in the prerequisite dependences
  all c : Course | c !in c.+prerequisites

} cond Realism {
  
  // Some prerequisites for courses being taken
  some c | some c.prerequisites && some c.enrolled

```
Academia Assertions

- A student can only wait to be in a course for which they have the prerequisites

```prolog
assert AllWaitsHavePrereqs { 
  all s | s.waitingfor.prerequisites in s.transcript 
}
```

For you to do (pause here)

- Load academia-2.all
- By looking at various sample instances, do you consider the model to be underconstrained in any way?
- With realism conditions enabled, do any instances exist in the default scopes?
  - Manipulate the scopes as necessary to obtain an instance under the realism conditions.
Analysis

- Instance generated
- With realism constraints default scopes is insufficient
- Check of assertion produces a counter-example

Counter-example

Analyzing AllWaitsHavePrereqs ...
Scopes: Course(3), Id(4), Person(4)
Counterexample found:
Domains:
  Course = \{C0,C1,C2\}     Id = \{I0,I1,I2,I3\}  Person = \{P0,P1,P2,P3\}
Sets:
  Faculty = \{\}            Graduate = \{P0,P1\}  Instructor = \{P0,P1\}
  Student = \{P0,P1,P2,P3\}                     Undergrad = \{P2,P3\}
Relations:
  enrolled = \{(C0 \rightarrow \{P0\}, C1 \rightarrow \{P2,P3\}, C2 \rightarrow \{P2,P3\}\}
  id = \{(P0 \rightarrow \{I3\}, P1 \rightarrow \{I2\}, P2 \rightarrow \{I1\}, P3 \rightarrow \{I0\}\}
  prerequisites = \{(C1 \rightarrow \{C0\}, C2 \rightarrow \{C0\}\}
  taughtby = \{(P0 \rightarrow \{P1\}, C1 \rightarrow \{P0\}, C2 \rightarrow \{P0\}\}
  teaches = \{(P0 \rightarrow \{C1,C2\}, P1 \rightarrow \{C0\}\}
  transcript = \{(P0 \rightarrow \{C0,C1,C2\}, P1 \rightarrow \{C1,C2\},
                P2 \rightarrow \{C0,C2\}, P3 \rightarrow \{C0,C1,C2\}\}
  waitingfor = \{(P1 \rightarrow \{C1,C2\}, P3 \rightarrow \{C0\}\}
  waitlist = \{(C0 \rightarrow \{P3\}, C1 \rightarrow \{P1\}, C2 \rightarrow \{P1\}\)
Extension 2

- Add Departments
  - Instructors per
  - Courses per
  - Required courses
  - Student majors
- Add Faculty-Grad student relationships
  - Advisor
  - Thesis committee

Department Associations

- Each instructor is in a single department.
  - Each department has at least one instructor.
- Each department has some courses
  - Courses are in a single department
- Each student has a single department as his/her major
  - i.e., a department
Faculty-Student Associations

- A graduate student has exactly one faculty member as an advisor.
- A faculty member serves on exactly one graduate student’s committee.

New Relations

```plaintext
state {
    ...
    department (~faculty) : Instructor+ -> Dept!
    courses : Dept! -> Course+
    required : Dept! -> Course+
    transcript : Student -> Course
    major (~majors) : Student -> Dept!
    advisor (~advisees) : Graduate -> Faculty!
    committee : Graduate -> Faculty
}
```

- At least one instructor per dept.
- Exactly one dept per instructor.
- Exactly one faculty member per graduate student.
New Constraints

- Advisors are on their student’s committees
- Students are advised by faculty in their major
- Only faculty teach required courses
- Required courses are a subset of the courses for a major
- Students must take at least one course from their major each semester
- Realism: There are at least two departments and some required courses.

Academia Constraints

inv {
  ...
  // advisors are always on their student's committees
  all g : Graduate | some g.committee -> g.advisor : g.committee
  // student's are advised by faculty in their major
  all g : Graduate | g.advisor : g.major.faculty
  // Only faculty are instructors for required courses
  all d : Dept | d.required.taughtby : Faculty
  // Faculty members only teach courses in their department
  all f : Faculty | f.teaches..~courses = f.department
  // Required courses for a major are courses in that major
  all d : Dept | d.required : d.courses
  // Students must take at least one course from their major each semester
  all s : Student | some (s.taking & s.major.courses)
}
cond Realism {
  ...
  // There are at least two majors
  some m : Dept | some (Dept - m)
  // Some required courses
  some Dept.required
}
A student’s committee members are faculty in his/her major.

```
assert CommitteeMembersInMajor {
    all s | s.committee.department in s.major
}
```

---

**For you to do**

- Load academia-3.all
- With realism conditions enabled, do any instances exist in the default scopes?
- Manipulate the scopes as necessary to obtain an instance under the realism conditions.
  - This requires some thought since a constraints may interact in subtle ways
  - For example, adding a department requires at least one faculty member for that department
- Can you think of any more questions about the model?
  - Formulate them as assertions and see if the properties are already enforced by the constraints.